Heuristic analysis
Jared Bowden

Question one: *Provide an optimal plan for Problems 1, 2, and 3.*

air_cargo_p1

```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

air_cargo_p2

```
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

air_cargo_p3

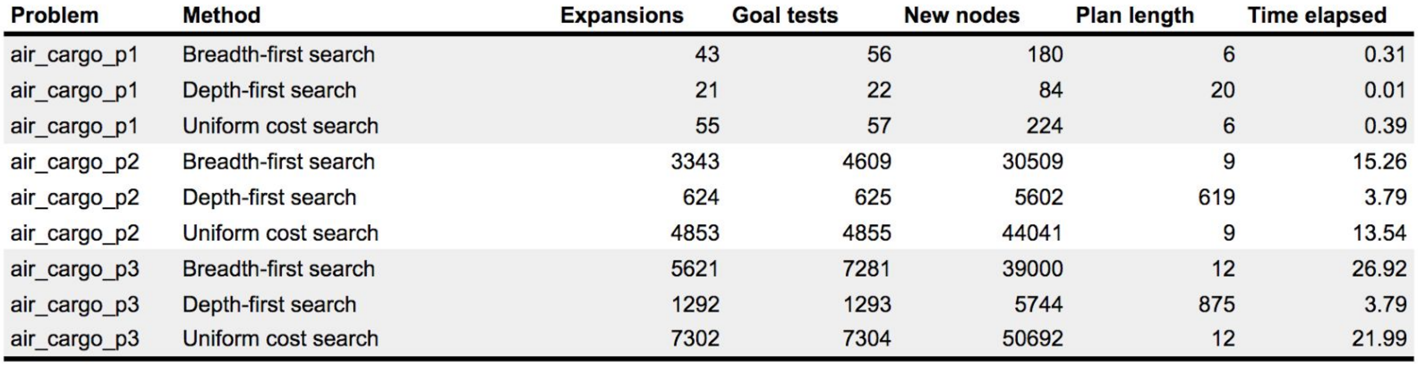
```
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Load(C2, P1, JFK)
Fly(P1, JFK, ORD)
Load(C4, P1, ORD)
Fly(P1, ORD, SFO)
Unload(C4, P1, SFO)
Unload(C2, P1, SFO)
```

Question two: *Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.*

The total path length of breadth-first search and uniform cost search was found to be comparable (and optimal) across problems one, two, and three (Table 1). Path length was uniformly highest for depth-first search. This result is unsurprising, given this algorithm’s function to explore *all* available paths (some of which are longer than optimal) before solving.

Total execution time was generally found to be higher for breadth-first search over uniform cost search; however, for the algorithms tested, the total number of nodes explored by uniform cost search was significantly higher for uniform cost search (Table 1). This too can be considered expected behavior, as uniform cost search is may expand unnecessarily when evaluating the cost of potential node paths.

Table 1. Comparison of results from uninformed non-heuristic search strategies: breadth-first search, depth-first search, and uniform cost search compared problems one, two, and three.



Problem	Method	Expansions	Goal tests	New nodes	Plan length	Time elapsed
air_cargo_p1	Breadth-first search	43	56	180	6	0.31
air_cargo_p1	Depth-first search	21	22	84	20	0.01
air_cargo_p1	Uniform cost search	55	57	224	6	0.39
air_cargo_p2	Breadth-first search	3343	4609	30509	9	15.26
air_cargo_p2	Depth-first search	624	625	5602	619	3.79
air_cargo_p2	Uniform cost search	4853	4855	44041	9	13.54
air_cargo_p3	Breadth-first search	5621	7281	39000	12	26.92
air_cargo_p3	Depth-first search	1292	1293	5744	875	3.79
air_cargo_p3	Uniform cost search	7302	7304	50692	12	21.99

Question two: raw data and methods

Methods

```
python run_search.py -p 1 -s 1  # breadth-first search
python run_search.py -p 1 -s 3  # depth-first search
python run_search.py -p 1 -s 5  # uniform cost search
```

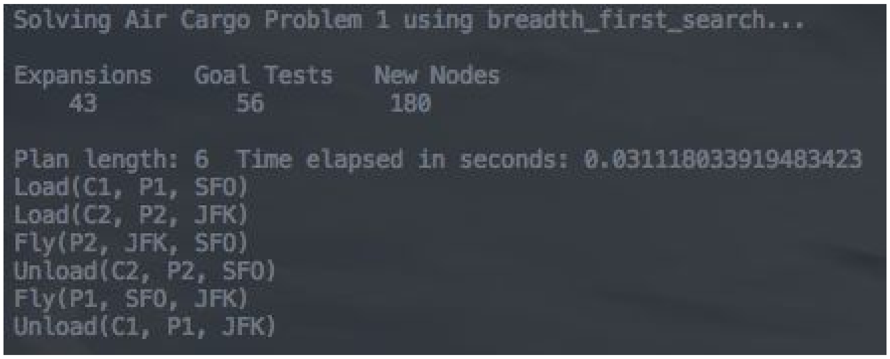
Script to run all (replace argument p to adapt for each problem)

```
python run_search.py -p 1 -s 1 && python run_search.py -p 1 -s 3 && python
run_search.py -p 1 -s 5
```

air_cargo_p1

Breadth-first search

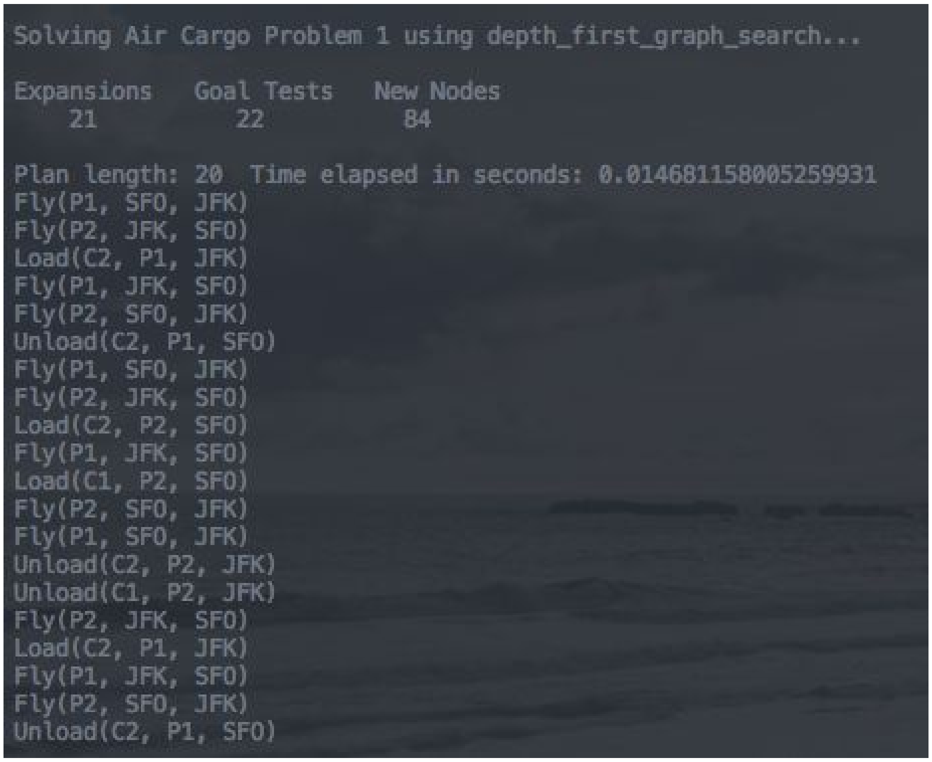
```
python run_search.py -p 1 -s 1
```



```
Solving Air Cargo Problem 1 using breadth_first_search...

Expansions   Goal Tests   New Nodes
    43           56          180

Plan length: 6  Time elapsed in seconds: 0.031118033919483423
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

Depth-first search [truncated results]

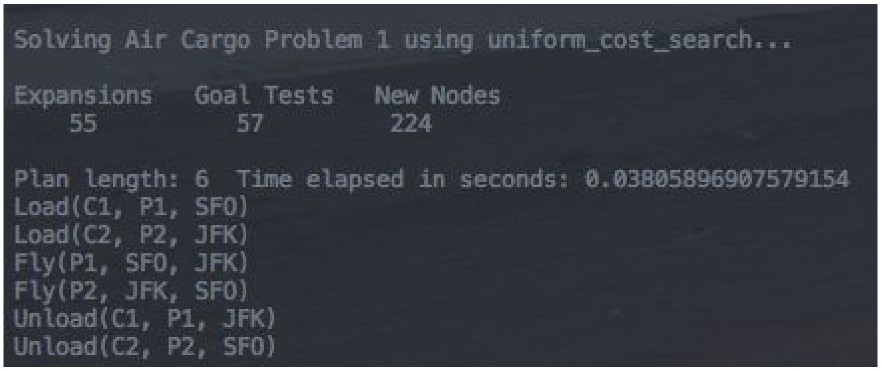
```
python run_search.py -p 1 -s 3
```



```
Solving Air Cargo Problem 1 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
    21           22          84

Plan length: 20  Time elapsed in seconds: 0.014681158005259931
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Load(C2, P1, JFK)
Fly(P1, JFK, SFO)
Fly(P2, SFO, JFK)
Unload(C2, P1, SFO)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Load(C2, P2, SFO)
Fly(P1, JFK, SFO)
Load(C1, P2, SFO)
Fly(P2, SFO, JFK)
Fly(P1, SFO, JFK)
Unload(C2, P2, JFK)
Unload(C1, P2, JFK)
Fly(P2, JFK, SFO)
Load(C2, P1, JFK)
Fly(P1, JFK, SFO)
Fly(P2, SFO, JFK)
Unload(C2, P1, SFO)
```

Uniform cost search

```
python run_search.py -p 1 -s 5
```



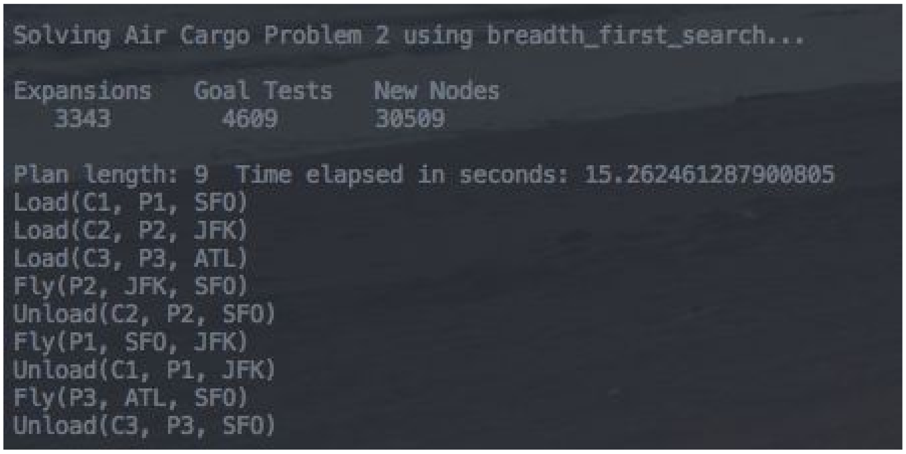
```
Solving Air Cargo Problem 1 using uniform_cost_search...

Expansions   Goal Tests   New Nodes
    55           57          224

Plan length: 6  Time elapsed in seconds: 0.03805896907579154
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

air_cargo_p2

```
python run_search.py -p 2 -s 1 && python run_search.py -p 2 -s 3  && python run_search.py -p 2 -s 5
```

Breadth-first search

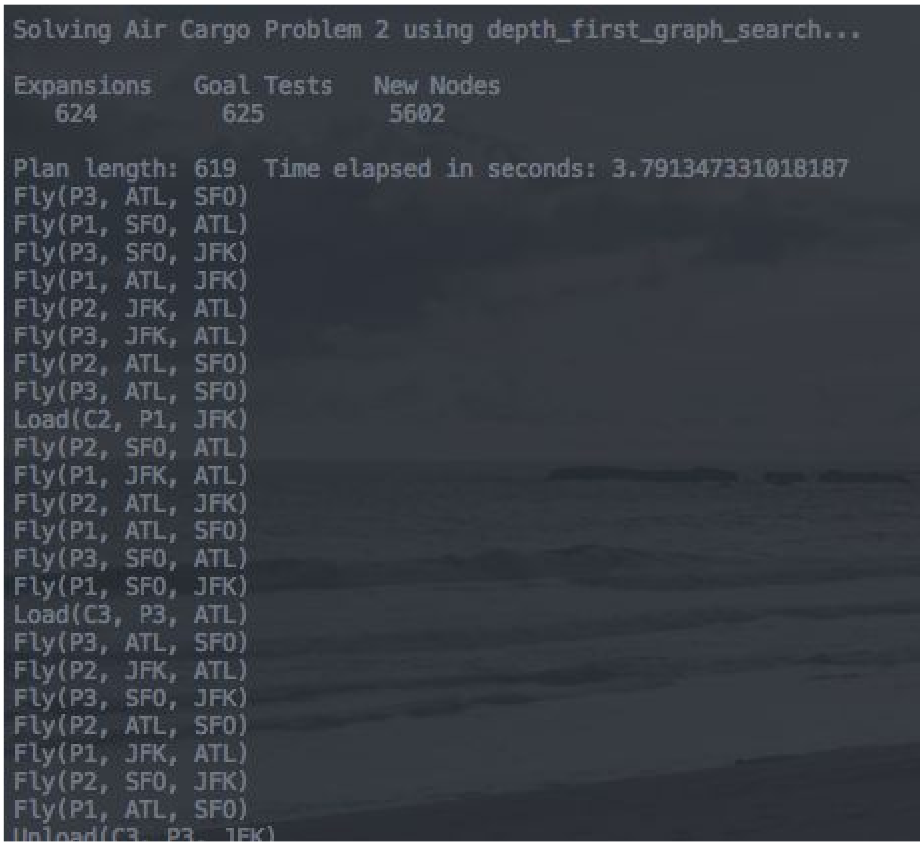
```
python run_search.py -p 2 -s 1
```



```
Solving Air Cargo Problem 2 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   3343         4609       30509

Plan length: 9  Time elapsed in seconds: 15.262461287900805
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
```

Depth-first search [truncated results]

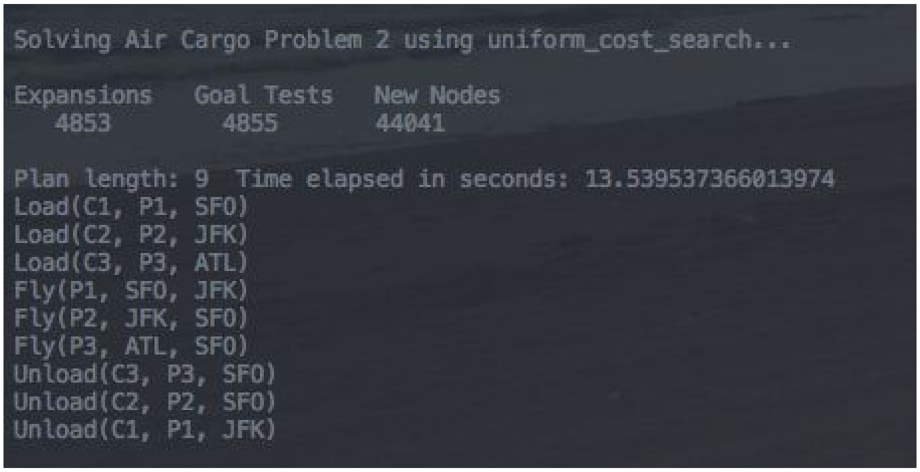
```
python run_search.py -p 2 -s 3
```



```
Solving Air Cargo Problem 2 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   624          625         5602

Plan length: 619  Time elapsed in seconds: 3.791347331018187
Fly(P3, ATL, SFO)
Fly(P1, SFO, ATL)
Fly(P3, SFO, JFK)
Fly(P1, ATL, JFK)
Fly(P2, JFK, ATL)
Fly(P3, JFK, ATL)
Fly(P2, ATL, SFO)
Fly(P3, ATL, SFO)
Load(C2, P1, JFK)
Fly(P2, SFO, ATL)
Fly(P1, JFK, ATL)
Fly(P2, ATL, JFK)
Fly(P1, ATL, SFO)
Fly(P3, SFO, ATL)
Fly(P1, SFO, JFK)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Fly(P2, JFK, ATL)
Fly(P3, SFO, JFK)
Fly(P2, ATL, SFO)
Fly(P1, JFK, ATL)
Fly(P2, SFO, JFK)
Fly(P1, ATL, SFO)
Unload(C3, P3, JFK)
```

Uniform cost search

```
python run_search.py -p 2 -s 5
```



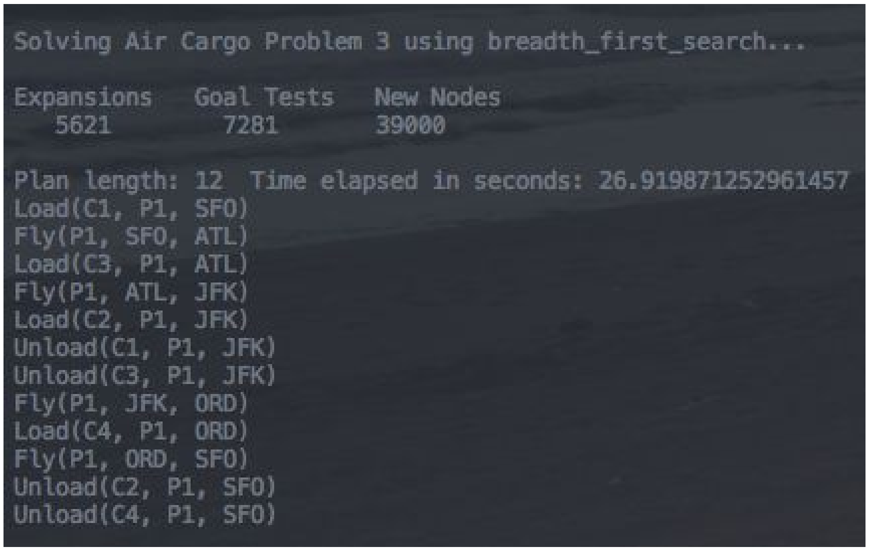
```
Solving Air Cargo Problem 2 using uniform_cost_search...

Expansions   Goal Tests   New Nodes
   4853         4855        44041

Plan length: 9  Time elapsed in seconds: 13.539537366013974
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

air_cargo_p3

```
python run_search.py -p 3 -s 1 && python run_search.py -p 3 -s 3 && python run_search.py -p 3 -s 5
```

Breadth-first search

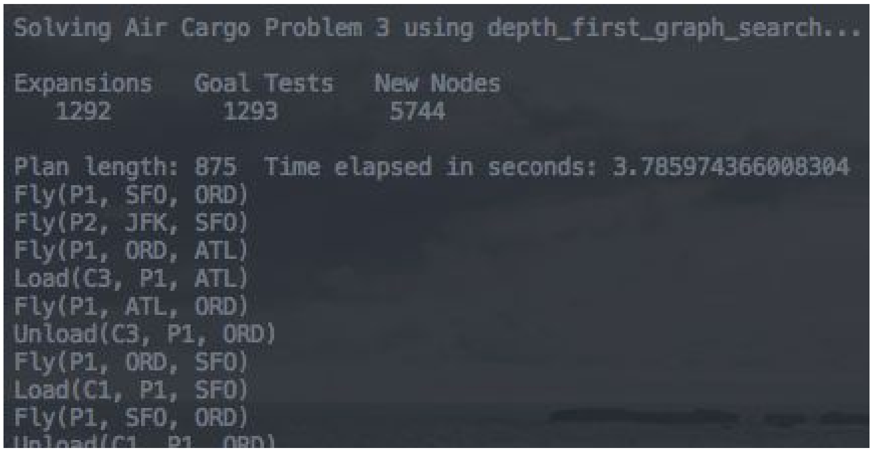
```
python run_search.py -p 3 -s 1
```



```
Solving Air Cargo Problem 3 using breadth_first_search...

Expansions   Goal Tests   New Nodes
   5621         7281        39000

Plan length: 12  Time elapsed in seconds: 26.919871252961457
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Load(C2, P1, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P1, JFK, ORD)
Load(C4, P1, ORD)
Fly(P1, ORD, SFO)
Unload(C2, P1, SFO)
Unload(C4, P1, SFO)
```

Depth-first search

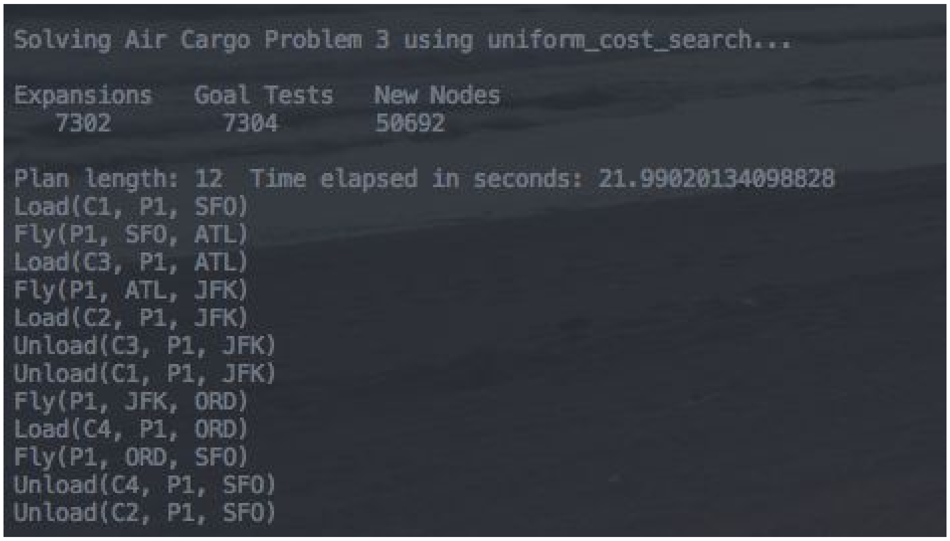
```
python run_search.py -p 3 -s 3
```



```
Solving Air Cargo Problem 3 using depth_first_graph_search...

Expansions   Goal Tests   New Nodes
   1292         1293         5744

Plan length: 875  Time elapsed in seconds: 3.785974366008304
Fly(P1, SFO, ORD)
Fly(P2, JFK, SFO)
Fly(P1, ORD, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, ORD)
Unload(C3, P1, ORD)
Fly(P1, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ORD)
Unload(C1, P1, ORD)
```

Uniform cost search

```
python run_search.py -p 3 -s 5
```

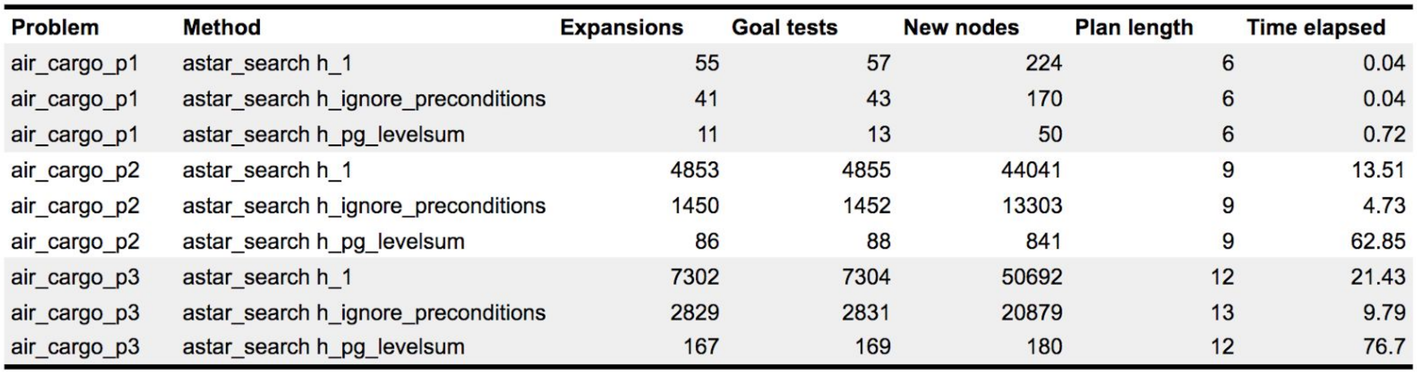


```
Solving Air Cargo Problem 3 using uniform_cost_search...

Expansions   Goal Tests   New Nodes
   7302         7304       50692

Plan length: 12  Time elapsed in seconds: 21.99020134098828
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Load(C2, P1, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Fly(P1, JFK, ORD)
Load(C4, P1, ORD)
Fly(P1, ORD, SFO)
Unload(C4, P1, SFO)
Unload(C2, P1, SFO)
```

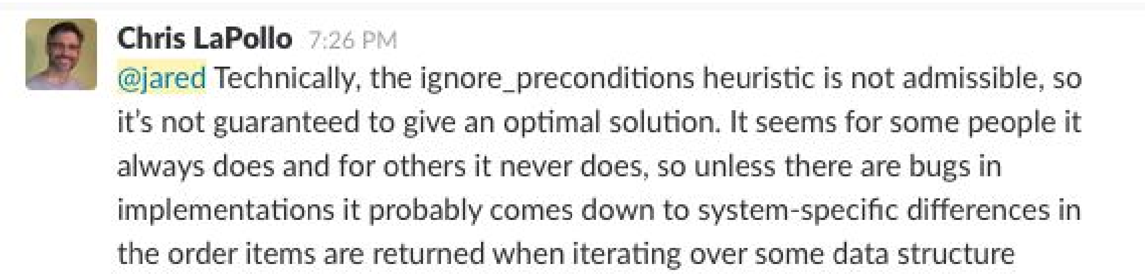
Question three: *Run A* planning searches using the heuristics you have implemented on `air_cargo_p1`, `air_cargo_p2` and `air_cargo_p3`. Provide metrics on number of node expansions required, number of goal tests, time elapsed, and optimality of solution for each search algorithm and include the results in your report.*

In general, heuristic informed A* search algorithms appeared to yielded optimal results (Table 2). One notable exception to this seems to have been the “ignore preconditions heuristic” on problem 3, which did not converge on an optimal solution (13 paths explored, as opposed to 12). I have an open comment discussing this on the AIND Slack channel (Supplementary Figure 1). Feedback serves to suggests this effect may be attributable to the fact that this heuristic application of A* does not guarantee a solution. The result, as it is currently show may reflect the order of expressions as they were initialized in the problem set. The “ignore preconditions heuristic” passes all unit tests locally, and in the AIND submission program.

Table 2. Comparison of results from A* search performance on problems one, two, and three, using 3 different heuristic methods.



Problem	Method	Expansions	Goal tests	New nodes	Plan length	Time elapsed
air_cargo_p1	astar_search h_1	55	57	224	6	0.04
air_cargo_p1	astar_search h_ignore_preconditions	41	43	170	6	0.04
air_cargo_p1	astar_search h_pg_levelsum	11	13	50	6	0.72
air_cargo_p2	astar_search h_1	4853	4855	44041	9	13.51
air_cargo_p2	astar_search h_ignore_preconditions	1450	1452	13303	9	4.73
air_cargo_p2	astar_search h_pg_levelsum	86	88	841	9	62.85
air_cargo_p3	astar_search h_1	7302	7304	50692	12	21.43
air_cargo_p3	astar_search h_ignore_preconditions	2829	2831	20879	13	9.79
air_cargo_p3	astar_search h_pg_levelsum	167	169	180	12	76.7



Chris LaPollo 7:26 PM

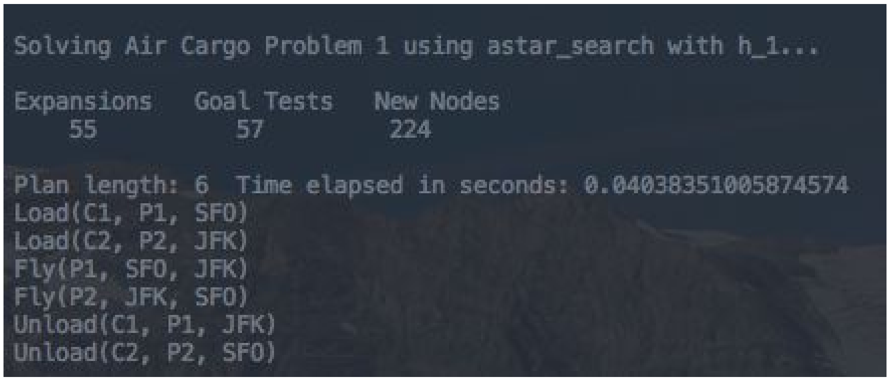
@jared Technically, the ignore_preconditions heuristic is not admissible, so it's not guaranteed to give an optimal solution. It seems for some people it always does and for others it never does, so unless there are bugs in implementations it probably comes down to system-specific differences in the order items are returned when iterating over some data structure

Supplementary Figure 1. *Addressing issues concerning optimality of the ignore preconditions heuristic in A* search.*

Question three: raw data and methods

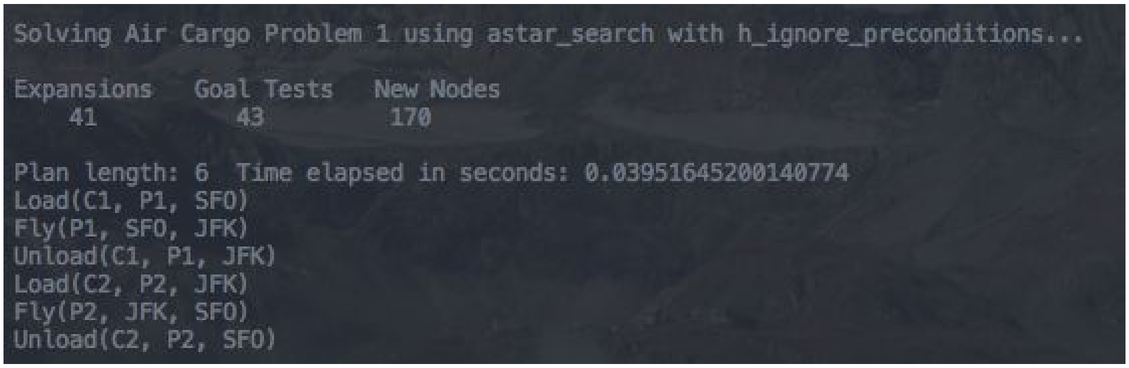
air_cargo_p1

```
python run_search.py -p 1 -s 8
```



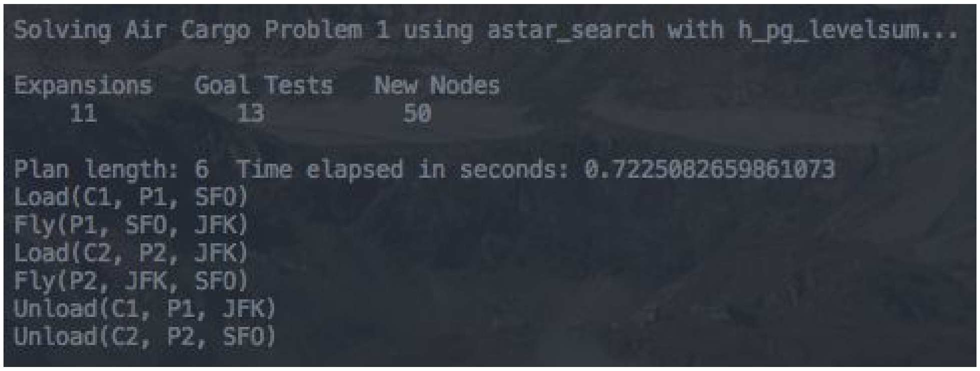
```
Solving Air Cargo Problem 1 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
    55           57         224

Plan length: 6  Time elapsed in seconds: 0.04038351005874574
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

```
python run_search.py -p 1 -s 9
```



```
Solving Air Cargo Problem 1 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
    41           43         170

Plan length: 6  Time elapsed in seconds: 0.03951645200140774
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

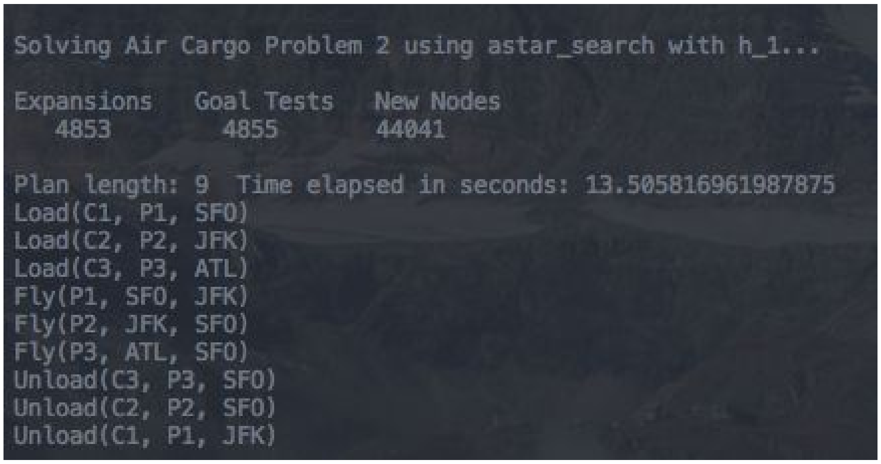
```
python run_search.py -p 1 -s 10
```



```
Solving Air Cargo Problem 1 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    11           13          50

Plan length: 6  Time elapsed in seconds: 0.7225082659861073
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

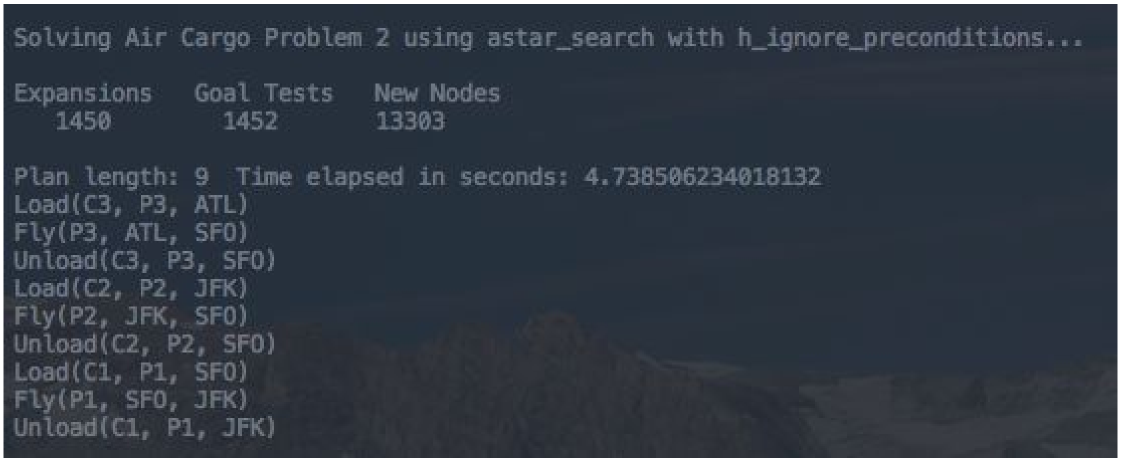
air_cargo_p2

```
python run_search.py -p 2 -s 8
```



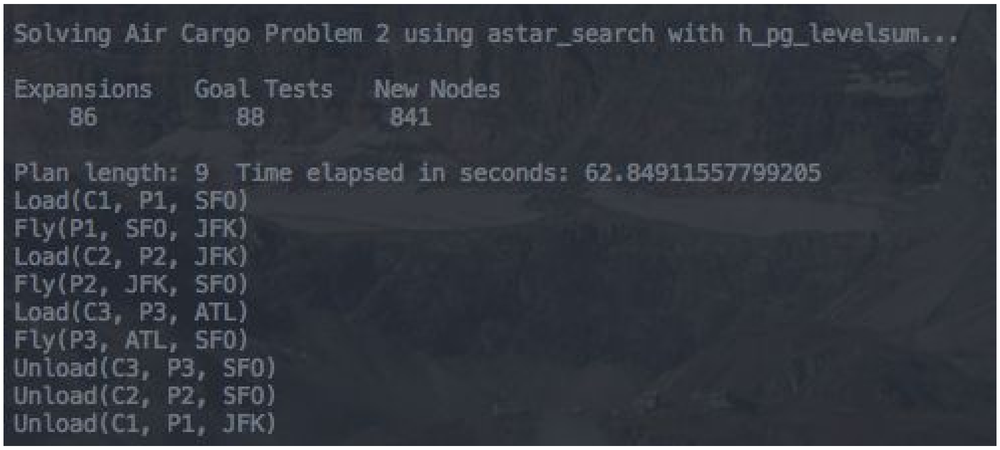
```
Solving Air Cargo Problem 2 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
   4853         4855       44041

Plan length: 9  Time elapsed in seconds: 13.505816961987875
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

```
python run_search.py -p 2 -s 9
```



```
Solving Air Cargo Problem 2 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   1450         1452       13303

Plan length: 9  Time elapsed in seconds: 4.738506234018132
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

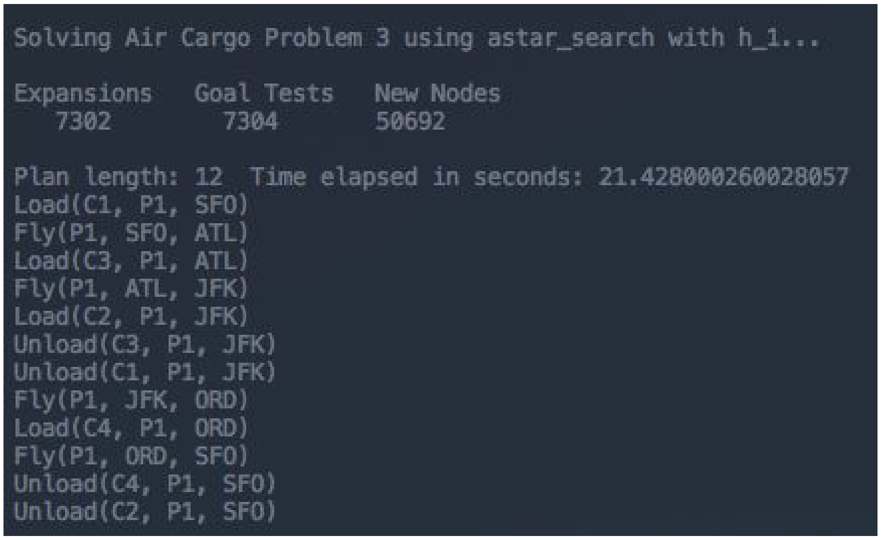
```
python run_search.py -p 2 -s 10
```



```
Solving Air Cargo Problem 2 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
    86           88         841

Plan length: 9  Time elapsed in seconds: 62.84911557799205
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

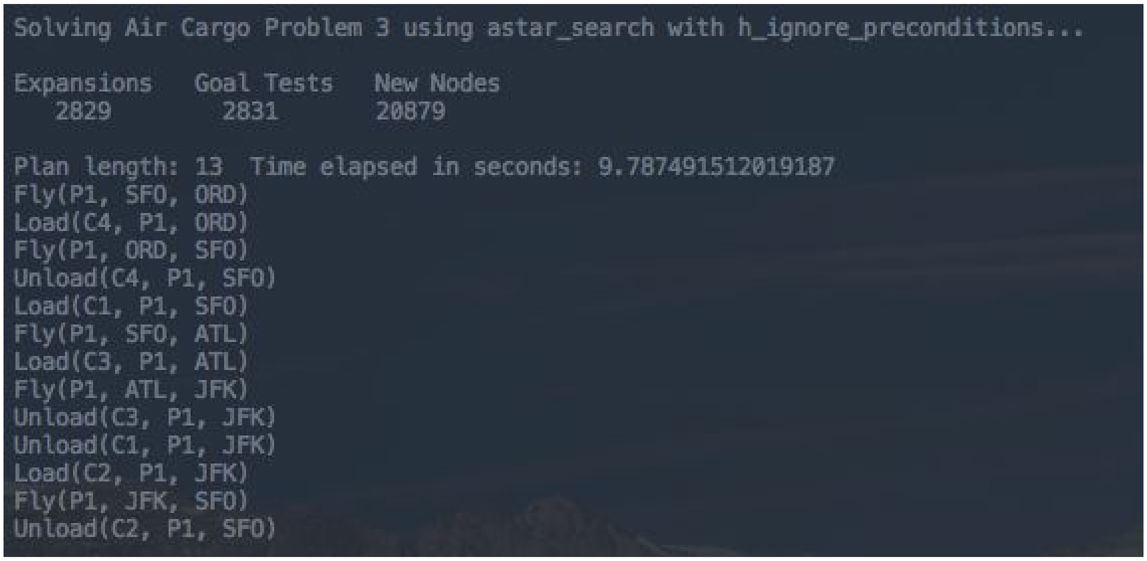
air_cargo_p3

```
python run_search.py -p 3 -s 8
```



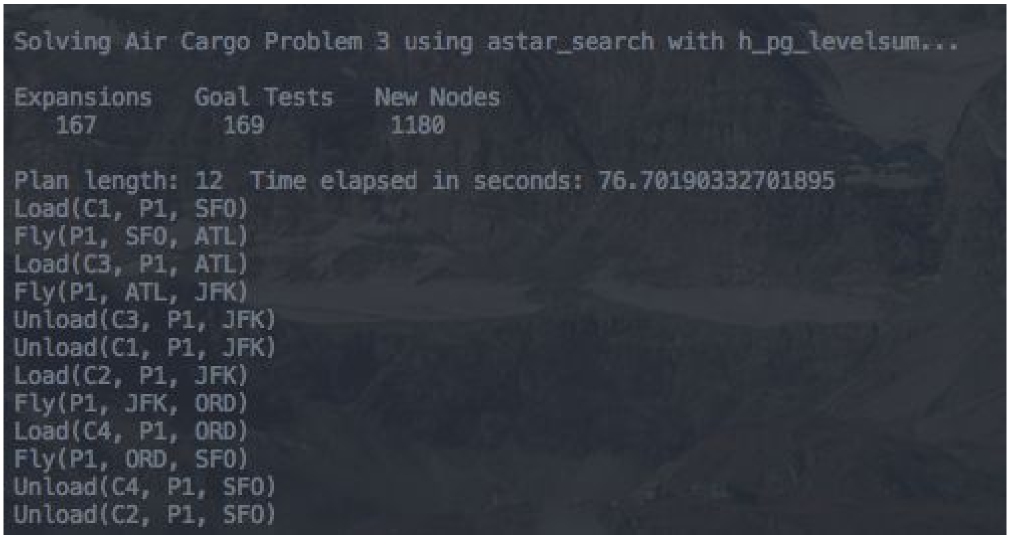
```
Solving Air Cargo Problem 3 using astar_search with h_1...

Expansions   Goal Tests   New Nodes
   7302        7304       50692

Plan length: 12  Time elapsed in seconds: 21.428000260028057
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Load(C2, P1, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Fly(P1, JFK, ORD)
Load(C4, P1, ORD)
Fly(P1, ORD, SFO)
Unload(C4, P1, SFO)
Unload(C2, P1, SFO)
```

```
python run_search.py -p 3 -s 9
```



```
Solving Air Cargo Problem 3 using astar_search with h_ignore_preconditions...

Expansions   Goal Tests   New Nodes
   2829        2831       20879

Plan length: 13  Time elapsed in seconds: 9.787491512019187
Fly(P1, SFO, ORD)
Load(C4, P1, ORD)
Fly(P1, ORD, SFO)
Unload(C4, P1, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Load(C2, P1, JFK)
Fly(P1, JFK, SFO)
Unload(C2, P1, SFO)
```

```
python run_search.py -p 3 -s 10
```



```
Solving Air Cargo Problem 3 using astar_search with h_pg_levelsum...

Expansions   Goal Tests   New Nodes
   167         169         1180

Plan length: 12  Time elapsed in seconds: 76.70190332701895
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Load(C2, P1, JFK)
Fly(P1, JFK, ORD)
Load(C4, P1, ORD)
Fly(P1, ORD, SFO)
Unload(C4, P1, SFO)
Unload(C2, P1, SFO)
```

Question 4 *What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?*

Almost all (see Question 3) of the heuristics tested in this A* search comparison converged on the optimal path length (Table 2); however, in terms of distinguishing features, the “ignore preconditions heuristic” appears to have achieved this result in the smallest amount of time.

In comparing the performance of heuristic and non-heuristic search algorithms, it’s fairly clear that heuristic search planning methods did not produce categorically superior results to non-heuristic methods (Table 1 and 2). In both cases, the processing required to arrive at a solution increased with the computational complexity of the problem, but overall average processing time for each problem was actually found to be *higher* for non-heuristic than heuristic methods.

One likely hypothesis resulting this finding may be that the problem space being tested here was not sufficiently complex as to warrant sophisticated methods like heuristic search planning. This effect is supported by the finding that the ignore preconditions heuristic, a reasonable simple approach, seemed to yield the most efficient results of the heuristic options available.

In conclusion, the current results suggest that that a carefully selected non-heuristic search method such as breadth-first search is capable of converging on an optimal solution for the air cargo problem more efficiently than more sophisticated heuristic applications of A* search.